(19) Eur pälsch s Patentamt
Eur pean Patent Office
Office européen des brev ts

(11) EP 0 751 458 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
04.10.2001 Bulletin 2001/40

(51) Int Cl.[7]: **G06F 9/38**

(21) Application number: **96480077.5**

(22) Date of filing: **31.05.1996**

(54) **Method and system for tracking resource allocation within a processor**

Verfahren und Anordnung zur Überwachung von Ressourcenzuteilung innerhalb eines Prozessors

Procédé et dispositif de suivi d'allocation des resources dans un processeur

(84) Designated Contracting States:
DE FR GB

(30) Priority: **29.06.1995 US 496833**

(43) Date of publication of application:
02.01.1997 Bulletin 1997/01

(73) Proprietor: **INTERNATIONAL BUSINESS MACHINES CORPORATION**
Armonk, NY 10504 (US)

(72) Inventor: **Chan, Kin**
Austin, Texas 78758 (US)

(74) Representative: **Therias, Philippe**
Compagnie IBM FRANCE,
Département de Propriété Intellectuelle
06610 La Gaude (FR)

(56) References cited:
EP-A- 0 312 239 **EP-A- 0 677 808**
US-A- 3 699 479

- **PROCEEDINGS, SUPERCOMPUTING '93, 15 November 1993, PORTLAND, OREGON, US, pages 636-644, XP000437401 J. K. PICKETT ET AL: "Enhanced Superscalar Hardware: The Schedule Table"**
- **IEEE TRANSACTIONS ON COMPUTERS, vol. 37, no. 5, May 1988, NEW YORK, NY, US, pages 562-573, XP000047779 J. E. SMITH ET AL: "Implementing Precise Interrupts in Pipelined Processors"**
- **PROCEEDINGS OF THE 26TH ANNUAL INTERNATIONAL SYMPOSIUM ON MICROARCHITECTURE, 1 - 3 December 1993, AUSTIN, TX, US, pages 202-213, XP000447502 M. MOUDGILL ET AL: "Register Renaming and Dynamic Speculation: an Alternative Approach"**

EP 0 751 458 B1

Description

BACKGROUND OF THE INVENTION

1. Technical Field:

[0001] The present invention relates in general to an improved method and system for data processing, and in particular to an improved method and system for tracking the allocation of resources within processor that supports speculative execution of instructions. Still more particularly, the present invention relates to a method and system for tracking the allocation of resources within a speculatively executing processor which enable the processor to recover the state of resource allocation following a mispredicted branch.

2. Description of the Related Art:

[0002] Designers of state-of-the-art processors are continually attempting to improve performance of such processors. Recently, processor designers have developed a number of architectural enhancements that have significantly improved processor performance over processors utilizing conventional architectures. For example, Reduced Instruction Set Computer (RISC) processors utilize reduced instruction sets that enable such processors to achieve low cycle-per-instruction (CPI) ratios. To further increase throughput, processors can also employ a superscaler architecture that enables multiple instructions to be issued and executed simultaneously by a number of execution units. As a further enhancement, execution units within a superscaler processor can be designed to execute in a pipelined fashion in which each execution unit processes multiple instructions simultaneously with one or more instructions at each stage of execution. Finally, state-of-the-art processors are equipped to execute instructions in an order determined by the availability of execution units rather than by sequential programmed order. This so-called "out of order" execution enables a processor to maximize the utilization of available execution unit resources during each cycle.

[0003] In a typical pipelined superscaler processor that supports out- of-order processing, one or more instructions are dispatched each cycle to a number of execution units. The instructions are executed opportunistically as execution unit resources become available with the caveat that the execution units must adhere to data dependencies between instructions. That is, if the execution of a first instruction depends upon data resulting from the execution of a second instruction, the first instruction must be executed prior to the second instruction. After an execution unit has completed processing an instruction, the instruction is forwarded to one of the number of completion buffers within the superscaler processor. A completion (rename) buffer is a temporary buffer which holds an instruction until the instruction is completed by transferring the data associated with the instruction from temporary registers to architected registers within the processor.

[0004] Although instructions can execute in any order as long as data dependencies are observed, most processors require that instructions are completed (i.e., data committed to architected registers) in program order. One reason for the requirement of in-order completion is to enable the processor to support precise interrupt and exception handling. For example, when an exception such as divide-by-zero arithmetic error occurs, an exception handler software routine will be invoked to manage the interrupt or exception. However, before the exception handler can be invoked, instructions preceding the instruction which generated the exception have to be completed in program order for the exception handler to execute in an environment that emulates the environment which would exist had the instructions been executed in program order. A second reason for the requirement of in-order completion is to enable proper recovery of a prior context if a branch is guessed wrong. As will be appreciated by those skilled in the art, superscaler processors typically include a branch execution unit, which predicts the result of branch instructions. Since the result of a branch instruction is guessed and instructions following the branch instruction reentry point are executed speculatively, the processor must have a mechanism for recovering a prior processor context if the branch is later determined to have been guessed wrong. Consequently, speculatively executed instructions cannot be completed until branch instructions preceding the speculatively executed instructions in program order have been completed.

[0005] In order to complete instructions executed out-of-order in program order, the processor must be equipped with facilities which track the program order of instructions during out-of-order execution. In conventional superscaler processors which support out-of-order execution, the program order of instructions is tracked by each of the execution units. However, as the number of execution units and the number of instructions which may be executed out-of-order increase, tracking the program order of instructions burdens the performance of the execution units. Consequently, it would be desirable to provide an improved method and system for managing the instruction flow within a superscaler processor which enables instructions to be dispatched in-order, executed out-of-order, and completed in-order and which does not require that the execution units track the program order of instructions.

[0006] A second source of performance problems within processors which support speculative execution of instructions is the recovery of the state of processor resources following a mispredicted branch. Typically, processors which support speculative execution of instructions include a branch history table (BHT) that enables a processor to predict the outcome of branch instructions based upon prior branch outcomes. Thus, uti-

lizing data within the BHT, the processor will begin execution of one or more sequential speculative execution paths which follow branch instruction reentry points. In conventional processors which support speculative execution, once a branch is determined to be guessed wrong, the processor stalls the execution pipeline until all sequential instructions preceding the misguessed branch are completed. Once all valid data is committed from the rename buffers to architected registers, all of the rename buffers are flushed and reset. Thereafter, the processor continues execution and allocation of the rename buffers beginning with the sequential instruction following the alternative execution path. Although this recovery mechanism guarantees that all of the processor resources will be available following a mispredicted branch, the conventional recovery mechanism degrades processor performance since the processor must delay dispatching additional instructions and allocating rename buffer resources until all instructions preceding the misguessed branch are completed.

[0007] Consequently, it would be desirable to provide an improved method and apparatus within a processor which enable the processor to restore the correct state of processor resources once a speculative execution path is determined to be mispredicted.

[0008] Publication entitled "Enhanced Superscalar Hardware : The Schedule Table" - 1993, 15 November 1993, Portland, Oregon, US pages 636-644, XP000437401 (J.K. Pickett et al) - examines a new technique for superscalar control implementation. In the push for ever increasing performance out of processor architectures, there is a need to expand beyond the limitations of existing scalar approaches. Superscalar architectures provide one of such means. By dynamically executing more than one instruction per clock cycle, superscalar architectures can improve performance without relying solely on technology improvements for these gains. This publication is directed to superscalar architectures and more particularly to a schedule table for facilitating dependency checking, out of order instruction issue, out of order execution, branch prediction, speculative execution, precise interrupts, and fast and efficient misprediction recovery.

SUMMARY OF THE INVENTION

[0009] It is therefore one object of the present invention to provide an improved method and system for data processing.

[0010] It is another object of the present invention to provide an improved method and system for tracking the allocation of resources within a processor that supports speculative execution of instructions.

[0011] It is yet another object of the present invention to provide an improved method and system for tracking the allocation of resources within a speculatively executing processor which enable the processor to recover the state of resource allocation following a mispredicted branch.

[0012] The foregoing objects are achieved as is now described. A method and system are disclosed for tracking the allocation of resources within a processor having multiple execution units which support speculative execution of instructions. The processor includes a resource counter including a first counter and a second counter and a number of resources, wherein one or more of the resources are allocated to each of a number of instructions dispatched for execution to the execution units. In response to dispatching an instruction among the plurality of instructions to one of the execution units for execution, the first counter is incremented once for each of the resources allocated to the instruction, and if the instruction is a first instruction within a speculative execution path, the second counter is loaded with a value of the first counter prior to incrementing the first counter. In response to completion of a particular instruction among the number of instructions dispatched to one of the multiple execution units, the first and the second counters are decremented once for each resource allocated to the particular instruction. In response to a refutation of the speculative execution path, a value of the second counter is transferred to the first counter, such that the resource counter tracks a number of the plurality of resources allocated to the plurality of instructions.

[0013] The above as well as additional objectives, features, and advantages of the present invention will become apparent in the following detailed written description.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 illustrates a preferred embodiment of a data processing system which utilizes the method and system of the present invention;

Figure 2 depicts a block diagram of the system unit of the data processing system illustrated in Figure 1;

Figure 3 illustrates a block diagram of a preferred embodiment of a processor which employs the method and system of the present invention;

Figure 4 depicts a more detailed block diagram of the instruction sequencing table (IST) illustrated in Figure 3;

Figure 5 illustrates a preferred embodiment of a counter which indicates a number of allocated en-

tries within the instruction sequencing table depicted in Figure 4;

Figure 6 depicts a preferred embodiment of a counter which indicates a number of allocated floating-point rename buffers;

Figure 7 illustrates a preferred embodiment of a counter which indicates a number of allocated general purpose rename buffers;

Figure 8 depicts a flowchart of the operation of the instruction sequencing table during a dispatch cycle;

Figure 9 illustrates a flowchart of the operation of the instruction sequencing table during a finish cycle; and

Figure 10 depicts a flowchart of the operation of the instruction sequencing table during a completion cycle.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENT

[0015] With reference now to the figures and in particular with reference to Figure 1, there is illustrated a block diagram of data processing system which employs the method and system of the present invention. As illustrated, data processing system 10 comprises system unit 12 and one or more local nodes 14, which include personal computer 16, display 18, keyboard 20, and mouse 22. As is well-known to those skilled in the art, a user inputs data to personal computer 16 utilizing keyboard 20, mouse 22, or other suitable input device. The user may then process the data locally utilizing personal computer 16, or transmit the data from personal computer 16 to system unit 12 or another node 14 utilizing well-known networking techniques. It is advantageous for a user to send tasks to system unit 12 for execution since system unit 12 can execute tasks in a relatively short period of time compared to node 14. System unit 12 and personal computer 16 output data to a user via display device 18.

[0016] Referring now to Figure 2, there is depicted a block diagram of system unit 12, which in a preferred embodiment of the present invention comprises a symmetric multiprocessor computer, such as the IBM RISC System/6000. System unit 12 includes one or more CPUs 30, which each include an on-board level one (L1) cache 32. Each CPU 30 is also associated with a level two (L2) cache 34. As will be understood by those skilled in the art, L1 caches 32 and L2 caches 34 each comprise a small amount of high-speed memory which store frequently accessed segments of data and instructions. If data requested by a CPU 30 is not resident within the L1 cache 32 or L2 cache 34 associated with CPU 30, the requested data is retrieved from main memory 36 via system bus 38.

[0017] System unit 12 also includes SCSI controller 40 and bus interface 46. SCSI controller 40 enables a user to attach additional SCSI devices 42 to system unit 12 via peripheral bus 44. Bus interface 46 provides facilities that enable multiple local nodes 14 to access system resources available within system unit 12. As will be appreciated by those skilled in the art, system unit 12 includes additional hardware coupled to system bus 46 that is not necessary for an understanding of the present invention and is accordingly omitted for simplicity.

[0018] With reference now to Figure 3, there is illustrated a preferred embodiment of a CPU 30 in accordance with the method and system of the present invention. In the preferred embodiment depicted in Figure 3, CPU 30 comprises a superscaler processor that issues multiple instructions into multiple execution pipelines each cycle, thereby enabling multiple instructions to be executed simultaneously. CPU 30 has five execution units 60-68, including fixed-point units 60 and 62, load-store unit 64, floating-point unit 66, and logical condition register unit 68.

[0019] According to the present invention, CPU 30 also includes instruction sequencing table (IST) 80, which enables CPU 30 to track the execution of instructions by execution units 60-68 and to complete instructions in program order. Referring now to Figure 4, there is depicted a block diagram of a preferred embodiment of IST 80. As illustrated, IST 80 includes a number of entries 110, which each contain a finish bit 112, exception code field 114, general purpose register (GPR) field 116, floating-point (FPR) register field 118, and branch bit 120. Entries 110 are addressed by one of 16 instruction IDs, which are each associated with an outstanding instruction, that is, an instruction that has been dispatched, but not completed.

[0020] With reference now to Figure 8, there is illustrated a flowchart of the operation of IST 80 during a dispatch cycle. As the process begins at block 200, instruction fetch address register (IFAR) 52 calculates the address of the next instructions to be fetched from instruction cache 54 based upon information received from program counter 104. The group of instructions specified by the address generated by IFAR 52 is loaded in parallel into instruction buffer 56 and dispatch unit 58 from instruction cache 54. The process then proceeds to block 202, which depicts determining a number of available entries 110 within IST 80. In a preferred embodiment of the present invention, the number of available entries 110 within IST 80 is easily determined from an IST entry counter 130 (illustrated in Figure 5) within resource counters 98 that counts the number of allocated IST entries 110. In the preferred embodiment illustrated in Figure 4, up to three instructions can be dispatched during each cycle if sufficient entries 110 are available within IST 80.

[0021] Next, at block 204, instruction buffer 56 reads out in program order a set of instructions for which IST entries 110 are available. Utilizing resource availability information received from completion unit 88 and resource counters 98, dispatch unit 58 enables selected ones of execution units 60-68 to begin execution of instructions for which resources, such as rename buffers 90 and 92, are available. Each instruction dispatched from instruction buffer 56 is assigned one of the instruction IDs specified by dispatch pointers 82. Since instructions are dispatched in program order, entries within IST 80 are allocated in program order. Thus, for the state of IST 80 depicted in Figure 4, if only a single instruction were dispatched during a dispatch cycle, that instruction would be assigned the entry 110 associated with instruction ID "1101" and specified as dispatch instruction ID 1 by dispatch pointers 82.

[0022] The process then proceeds to block 206, which illustrates writing completion information into IST 80 for each instruction dispatched. Each instruction issued from dispatch buffer 56 is processed by instruction decode unit (IDU) 70. IDU 70 decodes each instruction to determine the register resources required to complete the instruction. Thus, by determining the type of each instruction, IDU 70 can determine the number of general purpose registers (GPRs) and floating-point registers (FPRs) required to store the data associated with the instruction. Once IDU 70 has determined the register resources required to execute an instruction, IDU 70 writes the information into the appropriate entry 110 within IST 80. Next, the process proceeds to block 208, which depicts determining which, if any, of the dispatched instructions are speculative. If a dispatched instruction is the first instruction within a speculative execution path, the process proceeds to block 208, which depicts storing the dispatch pointer 82 (i.e., instruction ID) pointing to the entry allocated to the speculative instruction as a backup pointer 84. Storing the instruction ID of the first instruction within each speculative execution path enables CPU 30 to recover the correct execution context if a branch is later determined to have been misguessed.

[0023] The process proceeds from either block 208 or block 210 to block 212, which illustrates updating IST entry counter 130 and dispatch pointers 82. IST entry counter 130 is updated by IST control 100 which increments or decrements IST entry counter 130 by the net number of entries 110 allocated during the cycle after taking into account both dispatched and completed instructions. Dispatch pointers 82 are updated by incrementing the instruction ID to which dispatch pointers 82 point by the number of instructions dispatched during the cycle. Utilizing rotating pointers rather than a shifting queue enhances the performance of IST 80 since only dispatch pointers 82 are updated each cycle rather than everyentry 110. Thereafter, the process proceeds to block 214 where the process terminates.

[0024] Referring now to Figure 9, there is depicted a flowchart of the operation of IST 80 during a finish cycle. As is well known to those skilled in the art, each of execution units 60-68 is an execution pipeline having multiple stages, such as fetch, decode, execute, and finish, which can accommodate one or more instructions at each stage. Because execution units 60-68 operate independently and because the number of cycles required to execute instructions can vary due to data dependencies, branch resolutions, and other factors, execution units 60-68 execute instructions out of program order. As illustrated, the process begins at block 230 and thereafter proceeds to block 232, which depicts IST 80 receiving an instruction ID and finish report from execution units 60-68 for each instruction finished during the cycle. The finish report includes an exception code which identifies the exception generated by execution of the instruction, if any. The process then proceeds to block 234, which illustrates IST 80 writing the exception code received at block 232 into the exception code field 114 of the entry 110 identified by the finished instruction's ID. In addition, at block 234, finish bit 112 within entry 110 is set to indicate that the instruction has finished execution. In a preferred embodiment of the present invention, up to six finish reports can be written to IST 80 during a finish cycle. Following block 234, the process terminates at block 236.

[0025] With reference now to Figure 10, there is depicted a flowchart of the operation of IST 80 during a completion cycle. As illustrated, the process begins at block 240 and thereafter proceeds to block 242, which depicts completion unit 88 reading out instructions from IST 80 that are indicated by completion pointers 86. As depicted in Figure 4, a preferred embodiment of the present invention maintains three completion pointers 86 that specify instructions which can potentially be completed within a given processor cycle. The process then proceeds from block 242 to block 244, which illustrates completion unit 88 determining which of the instructions read out at block 242 generated exceptions that have not been handled. Completion unit 88 determines if an instruction generated an exception by examining the exception code field 114 associated with each instruction. If the first instruction (i.e. the instruction whose associated entry 110 is specified as completion instruction ID 1 by one of completion pointers 86) generated an exception, the process proceeds from block 244 through block 246 to block 248, which depicts forwarding the first instruction to interrupt handling unit 102. As will be understood by those skilled in the art, interrupt handling unit 102 calls an exception handling vector associated with the exception type specified by the exception code written within exception code field 114. Thereafter, the process proceeds from block 248 to block 254.

[0026] Returning to block 244, if the first instruction read out from IST 80 did not generate an exception, the process proceeds from block 244 through block 246 to block 249, which depicts determining which of the in-

structions read out at block 242 can be completed during the current cycle. In order to support precise interrupts, several constraints are placed upon the completion of instructions. First, only instructions that are marked as finished within IST 80 by finish bit 112 can be completed. Second, instructions that generated an exception which has not been handled cannot be completed in the present completion cycle. Third, an instruction can be completed only if all instructions preceding the instruction in program order have already been completed or will becompleted during the current completion cycle. Finally, for an instruction to be completed, the requisite number of general purpose registers and floating-point registers must be available within general purpose register file 94 and floating-point register file 96. Following block 249 the process proceeds to block 250, which depicts completion unit 88 completing instructions which satisfy the foregoing conditions by writing data associated with the instructions from GPR and FPR rename buffers 90 and 92 to GPR and FPR files 94 and 96.

[0027] Thereafter, the process proceeds from block 250 to block252, which depicts IST control 100 freeing IST entries 110 that are associated with the instructions completed at block 250. IST control 100 frees IST entries 110 by incrementing each of completion pointers 86 once for each instruction completed. Thereafter, the process proceeds to block 254 where the process terminates.

[0028] Referring now to Figures 5-7, there are illustrated block diagrams of IST entry counter 130, FPR rename buffer counter 150, and GPR rename buffer counter 170, which together comprise resource counters 98. With reference first to Figure 5, IST entry counter 130 includes multiplexers 132-137 and counters 138-142. According to a preferred embodiment of the present invention, counter 138 comprises a 17-bit shift counter which indicates in decoded format how many of the 16 IST entries 110 are currently allocated to outstanding instructions. Counter 138 is said to be in decoded format since the position of a set bit (a binary "1") within the counter indicates the number of allocated entries 110. For example, when IST 80 is empty, only the least significant (left most) bit is set, indicating that 0 entries 110 are allocated; if IST 80 is full , only the most significant bit is set. By storing the counters in decoded format rather than utilizing a register which is incremented and decremented by adders, the present invention not only minimizes the cycle time utilized to update counter 138, but also minimizes the complexity of CPU 30 and the chip substrate area consumed.

[0029] During each cycle IST control 100 computes the net change in the number of allocated entries 110 from the number of instructions dispatched and completed during that cycle. In a preferred embodiment of the present invention, the net change in the number of allocated entries 100 varies between +3 during cycles in which 3 instructions are dispatched and 0 instructions are completed to -3 during cycles in which 3 instructions are completed and 0 instructions are dispatched. IST control 100 updates counter 138 to reflect the current number of allocated entries 110 by selecting the appropriate update input to multiplexer 132, which in turn shifts the set bit within counter 138 a corresponding number of bit positions. Because an entry 110 is required for each instruction dispatched, counter 138 provides an interlock that prevents dispatch unit 58 from dispatching more instructions than can be accommodated within entries 110 in IST 80.

[0030] IST entry counter 130 also includes backup buffer counter A 140 and backup buffer counter B 142, which comprise shift counters like counter 138. Backup buffer counter A 140 indicates a number of allocated IST entries 110 excluding instructions within a first speculative execution path. Similarly, backup buffer counter B 142 indicates the number of allocated IST entries 110 excluding instructions within a second speculative execution path. As will be appreciated by those skilled in the art, embodiments of the present invention which support more than two speculative execution paths include one additional backup buffer counter for each additional speculative execution path permitted.

[0031] When the first instruction within a speculative execution path is dispatched, IST control 100 enables the select input to mux 133 to load the value of counter 138, which indicates the number of IST entries 110 allocated prior to dispatching instructions during the current cycle, into backup buffer counter A 140. In addition IST control 100 selects the appropriate update input to mux 134 to update backup buffer counter A 140. For example, if the second and third instructions dispatched are speculative and 3 outstanding instructions are completed during the current cycle, IST control 100 selects the -2 update input. Asillustrated, counter 140 can be incremented by a maximum of two entries since speculative instructions account for at least one of the three instructions which can be dispatched during the current cycle. During cycles while speculative execution path A remains unresolved, IST control logic 100 selects the appropriate path A input of mux 134 to update backup buffer counter A 140 to reflect the reduction in allocated entries 110 due to completion of outstanding nonspeculative instructions. If speculative execution path A is resolved as guessed correctly, the contents of backup buffer counter A 140 are simply ignored. If, however, speculative execution path A is resolved as guessed wrong, IST control 100 enables the select input to mux 137 to load the value of backup buffer counter A 140 into counter 138. In addition, IST control 100 selects the appropriate path A input to mux 132 to account for instructions completed during the current cycle. Thus, IST entry counter 138 maintains a correct count of allocated entries 110 even in cases where branches are misguessed.

[0032] As will be appreciated by those skilled in the art, mux 136 and backup buffer counter B 142 operate similarly to mux 134 and backup buffer counter A 140

to allow recovery from a second speculative execution path taken prior to the resolution of speculative path A. If speculative path A is resolved as correctly predicted and speculative path B (the second speculative execution path) is resolved as mispredicted, IST control 100 selects the appropriate input to mux 137 to load the value of backup buffer counter B 142 into counter 138. In addition, IST control 100 updates counter 138 by selecting the appropriate path B input to mux 132 to account for instructions completed during the current cycle.

[0033] Referring now to Figure 6, there is depicted a block diagram of FPR rename buffer counter 150, which indicates the number of allocated FPR rename buffers 92. As is evident by inspection of Figure 6, FPR rename buffer counter 150 functions much like IST entry counter 130. Backup buffer counter A 160 and backup buffer counter B 162 maintain a correct count of the number of allocated FPR rename buffers 92 in cases where either of two branch instructions are mispredicted, thereby enabling FPR rename buffer counter 150 to restore the correct FPR buffer count to counter 158 in a single cycle. In the illustrated embodiment, up to 3 of FPR rename buffers 92 can be assigned to instructions and up to 3 of FPR rename buffers 92 can be written to FPR file 96 during each cycle.

[0034] With reference now to Figure 7, there is illustrated a preferred embodiment of GPR rename buffer counter 170, which counts the number of GPR rename buffers 90 assigned to outstanding instructions. As will be appreciated by those skilled in the art, GPR rename buffer counter 170 operates similarly to FPR rename buffer counter 150, except for a difference in the number of GPR rename buffers 90 which can be allocated and retired within a cycle. In the depicted embodiment, up to two of GPR rename buffers 90 can be assigned to each instruction upon dispatch since two GPR rename buffers 90 are required to execute a "load and update" instruction. However, only two of GPR rename buffers 90 can be written to GPR file 94 during a given completion cycle.

[0035] The design of FPR and GPR rename buffer counters 150 and 170 enhances the performance of the present invention as compared to prior art systems since resources allocated to mispredicted branches can be more quickly reallocated. Prior art processors which support speculative execution of instructions typically do not include facilities such as backup buffer counters A and B to enable the processors to recover the correct state of processor resources following a misguessed branch. In conventional processors which support speculative execution, once a branch is determined to be guessed wrong, the processor stalls the execution pipeline until all sequential instructions preceding the misguessed branch are completed. Once all valid data is committed from the rename buffers to architected registers, all of the rename buffers are flushed and reset. Thereafter, the processor continues execution and allocation of the rename buffers beginning with the sequential instruction following the alternative execution path. Although this mechanism is relatively efficient in terms of the circuitry required to recover from a misguessed branch, the recovery mechanism degrades processor performance sinc the processor must delay dispatching additional instructions and llocating rename buffer resources until all instructions preceding the misguessed branch are completed.

[0036] As has been described, the present invention provides an improved method and system for managing the flow of instructions through a superscaler processor which supports out-of-order execution. By maintaining an entry corresponding to each outstanding instruction within an instruction sequencing table, the present invention enables instructions executed out-of-program order by multiple execution units to be completed in order, thereby supporting precise interrupts. Furthermore, the present invention provides an efficient mechanism for recovering from misguessed branches which enables the recovery of both the program state and the resource state of the processor prior to the misguessed branch. Although a processor which employs the present invention has been described with reference to various limitations with respect to a number of instructions which can be dispatched, finished, and completed during a given processor cycle, those skilled in the art will appreciate that these limitations are merely design choices and do not serve limitations on the present invention.

Claims

1. A method for tracking the allocation of resources within a processor (30) that supports speculative execution of instructions, said processor (30) having at least one execution unit (60-68), a resource counter (98) including a first counter (138, 158, 178) and a second counter (140/142, 160/162, 180/182) and a plurality of resources, wherein one or more of said plurality of resources are allocated to each of a plurality of instructions dispatched for execution to said plurality of execution units (60-68), said method comprising:

   in response to dispatching an instruction among said plurality of instructions to one of said at least one execution unit for execution:

   incrementing said first counter (138, 158, 178) once for each of said plurality of resources allocated to said dispatched instruction;

   if said dispatched instruction is a first instruction within a speculative execution path, loading said second counter (140/142, 160/162, 180/182) with a value of said first counter prior to incrementing said first counter;

in response to completion of a particular instruction among said plurality of instructions, decrementing said first and said second counters once for each resource allocated to said completed instruction; and

in response to resolution of said speculative execution path as mispredicted, transferring a value of said second counter to said first counter, wherein said resource counter tracks a number of said plurality of resources allocated to said plurality of instructions.

2. The method for tracking the allocation of resources within a processor of Claim 1, wherein said processor (30) comprises a superscalar processor capable of dispatching and completing multiple instructions during each cycle, wherein said step of loading said second counter with a value of said first counter further comprises:
incrementing said second counter once for each of said plurality of resources allocated to non-speculative instructions among said plurality of instructions that are dispatched concurrently with an instruction which is a first instruction within a speculative execution path.

3. The method for tracking the allocation of resources within a processor of Claim 1, said speculative execution path comprising a first speculative execution path, wherein said processor (30) concurrently supports a second speculative execution path and said resource counter further includes a third counter, said method further comprising:

in response to dispatching a selected instruction among said plurality of instructions to said at least one execution unit (60-68) for execution, wherein said selected instruction being a first instruction within a second speculative execution path, loading said third counter with a value of said first counter prior to incrementing said first counter;

in response to completion of a particular instruction among said plurality of dispatched instructions, decrementing said third counter once for each resource allocated to said particular instruction; and

in response to resolution of said first speculative execution path as correctly predicted and resolution of said second speculative execution path as mispredicted, transferring a value of said third counter to said first counter, wherein said resource counter tracks a number of said plurality of resources allocated to said plurality of instructions.

4. The method for tracking the allocation of resources within a processor of Claim 1, said first and said second counters comprising first and second shift registers, respectively, wherein each of said first and said second shift registers indicates a number of allocated resources among said plurality of resources by a bit position of a set bit, wherein said step of incrementing said first counter comprises shifting said set bit in a first direction within said first shift register one bit position for each of said plurality of resources allocated to said instruction, and wherein said step of decrementing said first and said second counters comprises shifting said set bits within said first and said second shift registers in a second direction one bit position for each resource allocated to said particular instruction.

5. An apparatus for tracking the allocation of resources within a processor (30) which supports speculative execution of instructions, said processor (30) having at least one execution unit (60-68) and a plurality of resources, wherein one or more of said plurality of resources are allocated to each of a plurality of instructions dispatched for execution to said at least one execution unit, said apparatus comprising:

a resource counter (98) having a first counter (138, 148, 178) and a second counter (140/142, 160/162, 180/182);

means, responsive to dispatching an instruction among said plurality of instructions to said at least one execution unit, for incrementing said first counter once for each of said plurality of resources allocated to said instruction;

means for loading said second counter with a value of said first counter prior to incrementing said first counter in response to dispatching an instruction among said plurality of instructions that is a first instruction within a speculative execution path;

means, responsive to completion of an instruction among said plurality of dispatched instructions, for decrementing said first and said second counters once for each resource allocated to said completed instruction; and

means for transferring a value of said second counter to said first counter in response to resolution of said speculative execution path as mispredicted, wherein said resource counter tracks a number of said plurality of resources allocated to said plurality of instructions.

6. The apparatus for tracking the allocation of resourc-

es within a processor (30) of Claim 5, wherein said processor (30) comprises a superscalar processor capable of dispatching and completing multiple instructions during each cycle, wherein said means for loading said second counter with a value of said first counter further comprises:
means for incrementing said second counter once for each of said plurality of resources allocated to non-speculative instructions among said plurality of instructions that are dispatched concurrently with an instruction which is said first instruction within said speculative execution path.

7. The apparatus for tracking the allocation of resources within a processor (30) of Claim 5, said speculative execution path comprising a first speculative execution path, wherein said processor (30) concurrently supports a second speculative execution path and said resource counter (98) further includes a third counter, said apparatus further comprising:

means for loading said third counter with a value of said first counter prior to incrementing said first counter prior to incrementing said first counter in response to dispatching a selected instruction among said plurality of instructions, wherein said selected instruction is a first instruction within a second speculative execution path;

means, responsive to completion of a particular instruction among said plurality of dispatched instructions, for decrementing said third counter once for each resource allocated to said particular instruction; and

means for transferring a value of said third counter to said first counter in response to resolution of a first speculative execution path as correctly predicted and resolution of said second speculative execution path as mispredicted, wherein said resource counter tracks a number of said plurality of resources allocated to said plurality of instructions; and

said first and said second counters comprising first and second shift registers, respectively, wherein each of said first and said second shift registers indicates a number of allocated resources among said plurality of resources by a bit position of a set bit within said first and said second shift registers, wherein said means for incrementing said first counter comprises means for shifting said set bit in a first direction within said first shift register one bit position for each of said plurality of resources allocated to said instruction, and wherein said means for decrementing said first and said second counters comprises means for shifting said set bits within said first and said second shift registers in a second direction one bit position for each resource allocated to said particular instruction.

8. The apparatus for tracking the allocation of resources within a processor (30) of Claim 5, wherein said plurality of resources comprise a plurality of rename data buffers (90, 92) utilized to store data associated with said plurality of instructions prior to completion; and wherein said processor supports out-of-order execution of said plurality of instructions and includes an instruction sequencing table (80) having a plurality of entries (110), wherein each of said plurality of instructions is assigned one of said plurality of entries sequentially according to a program order of said plurality of instructions, such that said plurality of instructions can be completed according to said program order, wherein said plurality of resources comprise said plurality of entries within said instruction sequencing table (80).

9. A processor (30), comprising:

at least one execution unit (60-68), wherein instructions dispatched to said at least one execution unit (60-68) can be executed out of program order;

a plurality of rename buffers (90, 92);

means for dispatching (58) instructions to said plurality of execution units (60-68);

means for assigning an instruction identifier (ID) to each of a plurality of instructions dispatched to said plurality of execution units (60-68) for execution, wherein an instruction identifier is assigned to each of said plurality of instructions sequentially according to a program order of said plurality of instructions;

a table (80) having a plurality of entries (110), wherein each entry (110) among said plurality of entries is associated with an instruction identifier (ID) and contains a finish indicator (112) that indicates whether execution of an instruction assigned an instruction identifier associated with said each entry has finished;

means for setting a finish indicator (112) within a particular entry (110) among said plurality of entries within said table (80) in response to termination of execution of an instruction assigned to an instruction identifier (ID) associated with said particular entry;

one or more pointers (86) which point to entries (110) within said table (80) associated with instruction identifiers (ID) assigned to a subset of said plurality of instructions that can possibly be completed during a particular processor cycle, wherein a selected instruction among said subset is completed by transferring data associated with said selected instruction from associated ones of said plurality of rename buffers (90, 92) to selected ones of said plurality of data registers; and

means for completing (88) selected instructions within said subset of said plurality of instructions, wherein exceptions generated by said selected instructions have been handled, wherein instructions among said plurality of instructions which are assigned instruction identifiers preceding said selected instructions have been completed during a previous processor cycle or will be completed during the same processor cycle, and wherein instruction identifiers assigned to said selected instructions are associated with entries having set finish indicators, such that said plurality of instructions are completed according to said program order.

10. The processor of Claim 9, wherein said processor further comprises a plurality of user-accessible data registers; and each entry (110) within said table (80) further comprises:

a field specifying a number of said plurality of data registers required to complete an instruction to which an instruction identifier (ID) associated with said each entry (110) is assigned; and

a field (114) indicating exception conditions which occurred during execution of an instruction to which an instruction identifier (ID) associated with said each entry (110) is assigned.

11. The processor of Claim 9, said table (80) having M entries (110), said processor further comprising:
an entry counter (130), including:

a primary shift register (138) having M+1 bits, wherein said primary shift register (138) indicates a first number of allocated entries among said plurality of entries (110) within said table (80) by a position of a set bit within said primary shift register;

a backup shift register (140) having M+1 bits, said backup shift register being associated with a speculative execution path, wherein said backup shift register indicates a second number of entries among said plurality of entries which are allocated to instructions that are not within said speculative execution path; and

means for transferring said second number from said backup shift register to said primary shift register in response to a determination that said speculative execution path was mispredicted; and

wherein said processor (30) supports N concurrent speculative execution paths, said entry counter (130) further comprising N backup shift registers (140, 142).

Patentansprüche

1. Verfahren zur Überwachung der Zuteilung von Ressourcen innerhalb eines Prozessors (30), der die spekulative Ausführung von Befehlen unterstützt, wobei der Prozessor (30) Folgendes aufweist: mindestens eine Ausführungseinheit (60 bis 68), einen Ressourcenzähler (98), der einen ersten Zähler (138, 158, 178) und einen zweiten Zähler (140/142, 160/162, 180/182) enthält, und eine Vielzahl von Ressourcen, wobei eine oder mehrere aus der Vielzahl von Ressourcen jedem aus einer Vielzahl von Befehlen zugeteilt werden, die der Vielzahl von Ausführungseinheiten (60 bis 68) zur Ausführung zugeteilt werden, wobei das Verfahren Folgendes umfasst:
auf die Zuteilung eines Befehls aus der Vielzahl von Befehlen zur Ausführung zu der mindestens einen Ausführungseinheit hin:

Erhöhen des ersten Zählers (138, 158, 178) einmal für jede aus der Vielzahl von Ressourcen, die dem zugeteilten Befehl zugeordnet werden;

Laden des zweiten Zählers (140/142, 160/162), 180/182) mit einem Wert des ersten Zählers vor dem Erhöhen des ersten Zählers, falls der zugeteilte Befehl ein erster Befehl in einem spekulativen Ausführungspfad ist;

Vermindern des ersten und zweiten Zählers einmal für jede Ressource, die dem ausgeführten Befehl zugeteilt wurde, auf die Ausführung eines bestimmten Befehls aus der Vielzahl von Befehlen hin; und

Übertragen eines Wertes des zweiten Zählers zum ersten Zähler auf die Auflösung des spekulativen Ausführungspfades hin, wenn er falsch vorhergesagt wurde, wobei der Ressour-

cenzähler eine Anzahl aus der Vielzahl von Ressourcen überwacht, die der Vielzahl von Befehlen zugeteilt werden.

2. Verfahren zur Überwachung der Zuteilung von Ressourcen innerhalb eines Prozessors nach Anspruch 1, wobei der Prozessor (30) einen Superskalarprozessor umfasst, der während jedes Zyklus mehrere Befehle zuteilen und ausführen kann, wobei der Schritt des Ladens des zweiten Zählers mit einem Wert des ersten Zählers außerdem Folgendes umfasst:
Erhöhen des zweiten Zählers einmal für jede aus der Vielzahl von Ressourcen, die nichtspekulativen Befehlen aus der Vielzahl von Befehlen zugeteilt werden, die gleichzeitig mit einem Befehl zugeteilt werden, der ein erster Befehl in einem spekulativen Ausführungspfad ist.

3. Verfahren zur Überwachung der Zuteilung von Ressourcen innerhalb eines Prozessors nach Anspruch 1, wobei der spekulative Ausführungspfad einen ersten spekulativen Ausführungspfad umfasst, wobei der Prozessor (30) gleichzeitig einen zweiten spekulativen Ausführungspfad unterstützt und der Ressourcenzähler außerdem einen dritten Zähler enthält, wobei das Verfahren außerdem Folgendes umfasst:

Laden des dritten Zählers mit einem Wert des ersten Zählers vor dem Erhöhen des ersten Zählers auf die Zuteilung eines ausgewählten Befehls aus der Vielzahl von Befehlen zur Ausführung zu der mindestens einen Ausführungseinheit (60 bis 68) hin, wobei der ausgewählte Befehl ein erster Befehl in einem zweiten spekulativen Ausführungspfad ist;

Vermindern des dritten Zählers einmal für jede dem bestimmten Befehl zugeteilte Ressource auf die Ausführung eines bestimmten Befehls aus der Vielzahl von zugeteilten Befehlen hin; und

Übertragen eines Wertes des dritten Zählers zum ersten Zähler auf die Auflösung des ersten spekulativen Ausführungspfades hin, wenn er korrekt vorhergesagt wurde, und auf die Auflösung des zweiten spekulativen Ausführungspfades hin, wenn er falsch vorhergesagt wurde, wobei der Ressourcenzähler eine Anzahl aus der Vielzahl von Ressourcen überwacht, die der Vielzahl von Befehlen zugeteilt werden.

4. Verfahren zur Überwachung der Zuteilung von Ressourcen innerhalb eines Prozessors nach Anspruch 1, wobei der erste und zweite Zähler erste bzw. zweite Schieberegister umfassen, wobei jedes der ersten und zweiten Schieberegister eine Anzahl von zugeteilten Ressourcen aus der Vielzahl von Ressourcen durch eine Bitposition eines gesetzten Bits anzeigt, wobei der Schritt des Erhöhens des ersten Zählers das Verschieben des gesetzten Bits in eine erste Richtung in eine Ein-Bit-Position des ersten Schieberegisters für jede aus der Vielzahl von dem Befehl zugeteilten Ressourcen umfasst, und wobei der Schritt des Verminderns des ersten und des zweiten Zählers das Verschieben der gesetzten Bits in den ersten und zweiten Schieberegistern in eine Ein-Bit-Position in einer zweite Richtung für jede dem bestimmten Befehl zugeteilten Ressource umfasst.

5. Vorrichtung zur Überwachung der Zuteilung von Ressourcen innerhalb eines Prozessors (30), der die spekulative Ausführung von Befehlen unterstützt, wobei der Prozessor (30) mindestens eine Ausführungseinheit (60 bis 68) und eine Vielzahl von Ressourcen aufweist, wobei eine oder mehrere aus der Vielzahl von Ressourcen jedem aus einer Vielzahl von Befehlen zugeteilt werden, die der mindestens einen Ausführungseinheit zur Ausführung zugeteilt werden, wobei die Vorrichtung Folgendes umfasst:

einen Ressourcenzähler (98), der einen ersten Zähler (138, 158, 178) und einen zweiten Zähler (140/142, 160/162, 180/182) aufweist;

Mittel zum Erhöhen des ersten Zählers einmal für jede aus der Vielzahl von dem Befehl zugeteilten Ressourcen, das auf die Zuteilung eines Befehls aus der Vielzahl von Befehlen zu der mindestens einen Ausführungseinheit anspricht;

Mittel zum Laden des zweiten Zählers mit einem Wert des ersten Zählers vor dem Erhöhen des ersten Zählers auf die Zuteilung eines Befehls aus der Vielzahl von Befehlen hin, der ein erster Befehl in einem spekulativen Ausführungspfad ist;

Mittel zum Vermindern der ersten und zweiten Zähler einmal für jede dem ausgeführten Befehl zugeteilten Ressource, das auf Ausführung eines Befehls aus der Vielzahl von zugeordneten Befehlen anspricht; und

Mittel zum Übertragen eines Wertes des zweiten Zählers zum ersten Zähler auf die Auflösung des spekulativen Ausführungspfades hin, wenn er falsch vorhergesagt wurde, wobei der Ressourcenzähler eine Anzahl aus der Vielzahl von Ressourcen überwacht, die der Vielzahl von Befehlen zugeteilt werden.

6. Vorrichtung zur Überwachung der Zuteilung von Ressourcen innerhalb eines Prozessors (30) nach Anspruch 5, wobei der Prozessor (30) einen Superskalarprozessor umfasst, der während jedes Zyklus mehrere Befehle zuteilen und ausführen kann, wobei das Mittel zum Laden des zweiten Zählers mit einem Wert des ersten Zählers außerdem Folgendes umfasst:
Mittel zum Erhöhen des zweiten Zählers einmal für jede aus der Vielzahl von Ressourcen, die nichtspekulativen Befehlen aus der Vielzahl von Befehlen zugeteilt werden, die gleichzeitig mit einem Befehl zugeteilt werden, der der erste Befehl im spekulativen Ausführungspfad ist.

7. Verfahren zur Überwachung der Zuteilung von Ressourcen innerhalb eines Prozessors (30) nach Anspruch 5, wobei der spekulative Ausführungspfad einen ersten spekulativen Ausführungspfad umfasst, wobei der Prozessor (30) gleichzeitig einen zweiten spekulativen Ausführungspfad unterstützt und der Ressourcenzähler (98) außerdem einen dritten Zähler enthält, wobei die Vorrichtung außerdem Folgendes umfasst:

Mittel zum Laden des dritten Zählers mit einem Wert des ersten Zählers vor dem Erhöhen des ersten Zählers auf die Zuteilung eines ausgewählten Befehls aus der Vielzahl von Befehlen hin, wobei der ausgewählte Befehl ein erster Befehl in einem zweiten spekulativen Ausführungspfad ist;

Mittel zum Vermindern des dritten Zählers einmal für jede dem bestimmten Befehl zugeteilten Ressource, das auf die Ausführung eines bestimmten Befehls aus der Vielzahl von zugeordneten Befehlen anspricht; und

Mittel zum Übertragen eines Wertes des dritten Zählers zum ersten Zähler auf die Auflösung des ersten spekulativen Ausführungspfades hin, wenn er korrekt vorhergesagt wurde, und auf die Auflösung des zweiten spekulativen Ausführungspfades hin, wenn er falsch vorhergesagt wurde, wobei der Ressourcenzähler eine Anzahl aus der Vielzahl von Ressourcen überwacht, die der Vielzahl von Befehlen zugeteilt werden; und

wobei der erste und zweite Zähler erste bzw. zweite Schieberegister umfassen, wobei jedes der ersten und zweiten Schieberegister eine Anzahl von zugeteilten Ressourcen aus der Vielzahl von Ressourcen durch eine Bitposition eines gesetzten Bits in den ersten und zweiten Schieberegistern anzeigt, wobei das Mittel zum Erhöhen des ersten Zählers Mittel zum Verschieben des gesetzten Bits in eine erste Richtung in der Ein-Bit-Position des ersten Schieberegisters für jede aus der Vielzahl von dem Befehl zugeteilten Ressourcen umfasst, und wobei das Mittel zum Vermindern des ersten und des zweiten Zählers Mittel zum Verschieben der gesetzten Bits in den ersten und zweiten Schieberegistern in eine Ein-Bit-Position in einer zweiten Richtung für jede dem bestimmten Befehl zugeteilten Ressource umfasst.

8. Vorrichtung zur Überwachung der Zuteilung von Ressourcen innerhalb eines Prozessors (30) nach Anspruch 5, wobei die Vielzahl von Ressourcen eine Vielzahl von Umbenennungsdatenpuffern (90, 92) umfassen, die zum Speichern der Daten verwendet werden, die der Vielzahl von Befehlen vor der Ausführung zugeordnet werden; und wobei der Prozessor eine Ausführung der Vielzahl von Befehlen außerhalb der Reihenfolge unterstützt und eine Befehlssortiertabelle (80) mit einer Vielzahl von Einträgen (110) enthält, wobei jeder aus der Vielzahl von Befehlen sequenziell gemäß einer Programmreihenfolge der Vielzahl von Befehlen einem aus der Vielzahl von Einträgen zugewiesen wird, so dass die Vielzahl von Befehlen gemäß der Programmreihenfolge ausgeführt werden können, wobei die Vielzahl von Ressourcen die Vielzahl von Einträgen in der Befehlssortiertabelle (80) umfassen.

9. Prozessor (30), Folgendes umfassend:

mindestens eine Ausführungseinheit (60 bis 68), wobei Befehle, die der mindestens einen Ausführungseinheit (60 bis 68) zugeteilt werden, außerhalb der Programmreihenfolge ausgeführt werden können;

eine Vielzahl von Umbenennungspuffern (90, 92);

Mittel zum Zuteilen (58) von Befehlen zu der Vielzahl von Ausführungseinheiten (60 bis 68);

Mittel zum Zuweisen einer Befehlskennung (ID) zu jedem aus einer Vielzahl von Befehlen, die der Vielzahl von Ausführungseinheiten (60 bis 68) zur Ausführung zugeteilt werden, wobei jedem aus der Vielzahl von Befehlen sequenziell gemäß einer Programmreihenfolge der Vielzahl von Befehlen eine Befehlskennung zugewiesen wird;

eine Tabelle (80), die eine Vielzahl von Einträgen (110) aufweist, wobei jedem Eintrag (110) aus der Vielzahl von Einträgen eine Befehlskennung (ID) zugeordnet wird, und die eine

Fertigstellungsanzeige (112) enthält, die anzeigt, ob die Ausführung eines Befehls fertiggestellt wurde, dem eine dem Eintrag zugeordnete Befehlskennung zugewiesen ist;

Mittel zum Setzen einer Fertigstellungsanzeige (112) in einem bestimmten Eintrag (110) aus der Vielzahl von Einträgen in der Tabelle (80) auf die Beendigung der Ausführung eines Befehls hin, dem eine dem bestimmten Eintrag zugeordnete Befehlskennung (ID) zugewiesen ist;

einen oder mehrere Zeiger (86), die auf Einträge (110) in der Tabelle (80) zeigen, die Befehlskennungen (ID) zugeordnet sind, die einem Teilsatz aus der Vielzahl von Befehlen zugewiesen sind, die während eines bestimmten Prozessorzyklus möglicherweise beendet werden können, wobei ein ausgewählter Befehl aus dem Teilsatz durch Übertragen von Daten zu ausgewählten aus der Vielzahl von Datenregistern beendet wird, wobei die Daten dem ausgewählten Befehl aus zugeordneten Puffern aus der Vielzahl von Umbenennungspuffern (90, 92) zugeordnet werden; und

Mittel zum Ausführen (88) von ausgewählten Befehlen in dem Teilsatz der Vielzahl von Befehlen, wobei von den ausgewählten Befehlen erzeugte Ausnahmen bearbeitet wurden, wobei Befehle aus der Vielzahl von Befehlen, denen Befehlskennungen zugewiesen werden, die den ausgewählten Befehlen vorausgehen, während eines vorhergehenden Prozessorzyklus ausgeführt wurden oder während desselben Prozessorzyklus ausgeführt werden, und wobei Befehlskennungen, die den ausgewählten Befehlen zugewiesen werden, Einträgen mit gesetzten Fertigstellungsanzeigen zugeordnet werden, so dass die Vielzahl von Befehlen gemäß der Programmreihenfolge ausgeführt werden.

10. Prozessor nach Anspruch 9, wobei der Prozessor außerdem eine Vielzahl von benutzerzugänglichen Datenregistern umfasst; und wobei jeder Eintrag in der Tabelle (80) außerdem Folgendes umfasst:

ein Feld, das eine Anzahl der Vielzahl von Datenregistern angibt, die zum Ausführen eines Befehls benötigt werden, dem eine jedem Eintrag (110) zugeordnete Befehlskennung (ID) zugewiesen wird; und

ein Feld (114), das Ausnahmebedingungen angibt, die während der Ausführung eines Befehls auftraten, dem eine jedem Eintrag (110) zugeordnete Befehlskennung (ID) zugewiesen wird.

11. Prozessor nach Anspruch 9, wobei die Tabelle (80) M Einträge (110) aufweist, wobei der Prozessor außerdem Folgendes umfasst:

einen Eintragszähler (130), der Folgendes enthält:

ein primäres Schieberegister (138) mit M+1 Bits, wobei das primäre Schieberegister (138) eine erste Anzahl von zugeteilten Einträgen aus der Vielzahl von Einträgen (110) in der Tabelle (80) durch eine Position eines gesetzten Bits im primären Schieberegister anzeigt;

ein Sicherungsschieberegister (140) mit M+1 Bits, wobei das Sicherungsschieberegister einem spekulativen Ausführungspfad zugeordnet wird, wobei das Sicherungsschieberegister eine zweite Anzahl von Einträgen aus der Vielzahl von Einträgen anzeigt, die Befehlen zugeteilt werden, die sich nicht im spekulativen Ausführungspfad befinden; und

Mittel zum Übertragen der zweiten Anzahl aus dem Sicherungsschieberegister zum primären Schieberegister auf eine Feststellung hin, dass der spekulative Ausführungspfad falsch vorhergesagt wurde; und

wobei der Prozessor (30) N gleichzeitig ablaufende spekulative Ausführungspfade unterstützt, wobei der Eintragszähler (130) außerdem N Sicherungsschieberegister (140, 142) umfasst.

Revendications

1. Procédé de suivi d'allocation des ressources dans un processeur (30) qui supporte l'exécution spéculative des instructions, ledit processeur (30) ayant au moins une partie exécution (60- 68), un compteur de ressources (98) comprenant un premier compteur (138, 158, 178) et un deuxième compteur (140/142, 160/162, 180/182) et une pluralité de ressources, où une ou plusieurs ressources de ladite pluralité de ressources sont allouées à chaque instruction d'une pluralité d'instructions distribuées en vue de leur exécution à ladite pluralité de parties exécutions (60-68), ledit procédé comprenant les phases qui consistent à :
en réponse à la distribution d'une instruction de ladite pluralité d'instructions à l'une des dites partie exécution pour qu'elle y soit exécutée :

augmenter d'un incrément la valeur dudit pre-

mier compteur (138, 158, 178) pour chacune des ressources de ladite pluralité de ressources qui allouée à ladite instruction distribuée ;

si ladite instruction distribuée est la première instruction dans un chemin d'exécution spéculative, charger dans ledit deuxième compteur (140/142, 160/162, 180/182) la valeur dudit premier compteur avant d'augmenter d'un incrément la valeur dudit premier compteur ;

en réponse à l'achèvement de l'exécution d'une instruction particulière parmi ladite pluralité d'instructions, diminue d'un incrément les valeurs desdits premier et deuxième compteurs pour chaque ressource allouée à ladite instruction achevée ; et

quand il est trouvé que ledit chemin d'exécution spéculative a été prédit incorrectement, transférer en réponse une valeur dudit deuxième compteur vers ledit premier compteur, où ledit compteur de ressources suit plusieurs desdites ressources allouées à ladite pluralité d'instructions.

2. Procédé de suivi d'allocation dès ressources dans un processeur selon la revendication 1, où ledit processeur (30) comprend un processeur super-scalaire pouvant distribuer et traiter de multiples instructions au cours de chaque cycle, où ladite phase qui consiste à charger dans ledit premier compteur une valeur dudit premier compteur comprend en outre la phase suivante :
augmenter d'un incrément la valeur dudit deuxième compteur pour chaque ressource de ladite pluralité de ressources allouées à des instructions non-spéculatives parmi ladite pluralité d'instructions qui sont distribuées en même temps qu'une instruction qui est la première instruction dans un chemin d'exécution spéculative.

3. Procédé de suivi d'allocation des ressources dans un processeur selon la revendication 1, ledit chemin d'exécution spéculative comprenant un premier chemin d'exécution spéculative, où ledit processeur (30) supporte en même temps un deuxième chemin d'exécution spéculative et ledit compteur de ressources comprend en outre un troisième compteur, ledit procédé comprenant également les phases suivantes :

en réponse à la distribution à ladite (aux dites) partie(s) exécution (60-68) d'une instruction choisie parmi ladite pluralité d'instructions en vue de son exécution, où ladite instruction choisie est la première instruction dans un deuxième chemin d'exécution spéculative, charger dans ledit troisième compteur une valeur dudit premier compteur avant d'augmenter d'un incrément la valeur dudit premier compteur ;

en réponse à l'achèvement de l'exécution d'une instruction particulière parmi ladite pluralité d'instructions distribuées, diminuer d'un incrément la valeur dudit troisième compteur pour chaque ressource allouée à ladite instruction particulière ; et

quand il est trouvé que ledit premier chemin d'exécution spéculative a été prédit correctement et qu'il est trouvé que ledit deuxième chemin d'exécution spéculative a été prédit de manière incorrecte, transférer en réponse une valeur dudit troisième compteur vers ledit premier compteur, où ledit compteur de ressources suit plusieurs des dites ressources allouées à ladite pluralité d'instructions.

4. Procédé de suivi d'allocation de ressources dans un processeur selon la revendication 1, ledit premier et ledit deuxième compteurs comprenant respectivement un premier et un deuxième registres de décalage, où chacun desdits premier et deuxième registres de décalage indique un nombre de ressources allouées parmi ladite pluralité de ressources par la position binaire d'un bit de réglage, où ladite phase d'accroissement d'un incrément de la valeur dudit premier compteur comprend la phase qui consiste à décaler dans un premier sens dans ledit premier registre de décalage ledit bit de réglage, d'une position binaire pour chaque ressource de ladite pluralité de ressources allouées à ladite instruction, et où ladite phase de décrémentation desdites premier et deuxième compteurs comprend de décaler, d'une position binaire pour chaque ressource allouée à ladite instruction particulière, lesdits bits de réglage dans lesdits premier et deuxième registres de décalage dans une deuxième direction.

5. Appareil pour suivre l'allocation des ressources dans un processeur (30) qui supporte l'exécution spéculative des instructions, ledit processeur (30) ayant au moins une partie exécution (60-68) et une pluralité de ressources, où une ou plusieurs ressources de ladite pluralité de ressources sont allouées à chaque instruction d'une pluralité d'instructions distribuées à ladite pluralité de parties exécutions pour être exécutées, ledit appareil comprenant :

un compteur de ressources (98) comportant un premier compteur (138, 148, 178) et un deuxième compteur (140/142, 160/162, 180/182) ;

un moyen qui, en réponse à la distribution d'une instruction de ladite pluralité d'instructions à ladite (ou auxdites) partie(s) exécution, augmente d'un incrément la valeur dudit premier compteur pour chacune des ressources de ladite pluralité de ressources allouées à ladite instruction ;

un moyen pour charger dans ledit deuxième compteur la valeur dudit premier compteur avant l'accroissement d'un incrément de la valeur dudit premier compteur, en réponse à la distribution d'une instruction de ladite pluralité d'instructions qui est une première instruction dans un chemin d'exécution spéculative ;

un moyen qui, en réponse à l'achèvement de l'exécution d'une instruction de ladite pluralité d'instructions distribuées, diminue d'un incrément la valeur desdits premier et deuxième compteurs pour chaque ressource allouée à ladite instruction dont l'exécution est terminée ; et

un moyen pour, en réponse, transférer la valeur dudit deuxième compteur vers ledit premier compteur quand il a été trouvé que le chemin d'exécution spéculative prédit était incorrect, où ledit compteur de ressources suit de nombreuses ressources de ladite pluralité de ressources allouées à ladite pluralité d'instructions.

6. Appareil pour suivre l'allocation de ressources dans un processeur (30) selon la revendication 5, dans lequel ledit processeur (30) comprend un processeur superscalaire pouvant distribuer et traiter de multiples instructions au cours de chaque cycle, où ledit moyen pour charger dans ledit deuxième compteur une valeur dudit premier compteur comprend en outre :
un moyen pour augmenter d'un incrément la valeur dudit deuxième compteur pour chaque ressource de ladite pluralité de ressources allouées à des instructions non spéculatives qui, parmi ladite pluralité d'instructions, sont distribuées en même temps qu'une instruction qui est ladite première instruction dans ledit chemin d'exécution spéculative.

7. Appareil pour suivre l'allocation des ressources dans un processeur (30) selon la revendication 5, ledit chemin d'exécution spéculative comprenant un premier chemin d'exécution spéculative, où ledit processeur (30) supporte en même temps un deuxième chemin d'exécution spéculative et ledit compteur de ressources (98) comprend en outre un troisième compteur, ledit appareil comprenant également :

un moyen pour charger dans ledit troisième compteur une valeur dudit premier compteur avant d'augmenter d'un incrément la valeur dudit premier compteur avant d'augmenter d'un incrément la valeur dudit premier compteur en réponse à la distribution d'une instruction choisie parmi ladite pluralité d'instructions, où ladite instruction choisie est la première instruction dans un deuxième chemin d'exécution spéculative ;

un moyen qui, en réponse à l'achèvement de l'exécution d'une instruction particulière parmi ladite pluralité d'instructions distribuées, diminue d'un incrément la valeur dudit troisième compteur pour chaque ressource allouée à ladite instruction particulière ; et

un moyen pour répondre en transférant une valeur dudit troisième compteur vers ledit premier compteur quand il est trouvé qu'un premier chemin d'exécution spéculative a correctement été prédit et que ledit deuxième chemin d'exécution spéculative n'a pas été prédit correctement, où ledit compteur de ressources suit plusieurs ressources de ladite pluralité de ressources allouées à ladite pluralité d'instructions ; et

lesdits premier et deuxième compteurs comprenant respectivement des premier et deuxième registres de décalage, où chacun desdits premier et deuxième registres de décalages désigne plusieurs ressources allouées parmi ladite pluralité de ressources, par la position binaire d'un bit de réglage dans lesdits premier et deuxième registres de décalage, où ledit moyen pour incrémenter la valeur dudit premier compteur comprend un moyen pour décaler ledit bit de réglage dans une première direction, dans ledit premier registre de décalage, d'une position binaire pour chacune desdites ressources allouées à ladite instruction, et où ledit moyen pour décrémenter ledit premier compteur et ledit deuxième compteur comprend un moyen pour décaler lesdits bits de réglage, à l'intérieur desdits premier et deuxième registres de décalage, d'une position binaire dans une deuxième direction pour chaque ressource allouée à ladite instruction particulière.

8. Appareil pour suivre l'allocation de ressources dans un processeur (30) selon la revendication 5, dans lequel ladite pluralité de ressources comprend une pluralité de tampons de données de redésignation (90, 92) utilisés pour enregistrer les données associées à ladite pluralité d'instructions avant qu'elles aient été entièrement exécutées ; et où ledit processeur supporte l'exécution dans le désordre de ladite

pluralité d'instructions et inclut une table de classement des instructions (80) comportant une pluralité d'entrées (110) où, à chaque instruction de ladite pluralité d'instructions est assignée séquentiellement une de ladite pluralité d'entrées, suivant un ordre du programme composé par ladite pluralité d'instructions, pour que ladite pluralité d'instructions puisse être entièrement exécutée suivant ledit ordre de programme, où ladite pluralité de ressources comprend ladite pluralité d'entrées inscrites dans ladite table de séquencement des instructions (80).

9. Processeur (30) comprenant :

au moins une partie exécution (60-68), où les instructions distribuées à ladite (ou auxdits) partie exécution (60-68) peuvent être exécutées dans le désordre ;

une pluralité de tampons de redésignation (90, 92) ;

un moyen pour distribuer (58) les instructions à ladite pluralité de parties exécution (60-68) ;

un moyen pour assigner un identificateur d'instruction (ID) à chaque instruction d'une pluralité d'instructions distribuées à ladite pluralité de parties exécution (60-68) pour être exécutées, où un identificateur d'instruction est assigné séquentiellement à chaque instruction de ladite pluralité d'instructions, suivant l'ordre du programme composé par ladite pluralité d'instructions ;

une table (80) ayant une pluralité d'entrées (110), où chaque entrée (110) parmi ladite pluralité d'entrées est associée à un identificateur d'instruction (ID) et contient un indicateur de fin (112) qui indique si l'exécution d'une instruction à laquelle est assigné un identificateur associé à chacune desdites entrées est terminée ;

un moyen pour régler un indicateur de fin (112) dans une entrée particulière (110) parmi ladite pluralité d'entrées dans ladite table (80), en réponse à l'achèvement de l'exécution d'une instruction assignée à un identificateur d'instruction (ID) associé à ladite entrée particulière ;

un ou plusieurs index (86) qui désignent les entrées (110) dans la table (80) qui sont associées à des identificateurs d'instructions (ID) assignés à un sous-ensemble de ladite pluralité d'instructions qui pourraient être entièrement être exécutées pendant un cycle particulier, où une instruction choisie dudit sous-ensemble est entièrement traitée par le transfert des données qui sont associées à cette dite instruction choisie, depuis certains tampons associés de ladite pluralité de tampons de redésignation (90, 92) vers certains registres sélectionnés par mi ladite pluralité des registres de données ; et

un moyen pour terminer (88) des instructions choisies dudit sous-ensemble de ladite pluralité d'instruction, où les exceptions générées par lesdites instructions choisies ont été gérées, où les instructions, parmi ladite pluralité d'instructions, auxquelles sont assignés des identificateurs d'instruction et qui précèdent lesdites instructions choisies ont terminé d'être exécutées au cours d'un cycle précédent ou seront achevées au cours du même cycle, et où les identificateurs d'instruction assignés auxdites instructions choisies sont associés à des entrées dont les indicateurs de fin sont réglés, de sorte que ladite pluralité d'instructions est entièrement exécutée selon ledit ordre de programme.

10. Processeur selon la revendication 9, où ledit processeur comprend en outre une pluralité de registres de données accessibles aux utilisateurs ; et chaque entrée (110) dans ladite table (80) comprend en outre :

un champ spécifiant un nombre de ladite pluralité de registres de données requis pour traiter entièrement une instruction à laquelle est assigné un identificateur d'instruction (ID) associé à ladite entrée (110) ; et

un champ (114) indiquant les cas d'exception qui se sont produits au cours de l'exécution d'une instruction à laquelle un identificateur d'instruction (ID) associé à ladite entrée (110) est assigné.

11. Processeur selon la revendication 9, ladite table (80) comportant M entrées (110), ledit processeur comprenant en outre :

un compteur d'entrées (130), comprenant :

un registre de décalage principal (138) ayant M+1 bits, où ledit registre de décalage principal (138) indique un premier nombre d'entrées allouées parmi ladite pluralité d'entrées (110) dans ladite table (80) par la position d'un bit de réglage dans ledit registre de décalage principal ;
un registre de décalage de secours (140) ayant M+1 bits, ledit registre de décalage de secours étant associé à un chemin d'exécution spéculative, où ledit registre de décalage de secours

indique un deuxième nombre d'entrées, parmi ladite pluralité d'entrées, qui sont allouées à des instructions qui ne sont pas dans ledit chemin d'exécution spéculative ; et

un moyen pour transférer, en réponse, ledit deuxième nombre dudit registre de décalage de secours vers ledit registre de décalage principal quand il est trouvé que ledit chemin d'exécution spéculative n'a pas été prédit correctement ; et

où ledit processeur (30) supporte N chemins d'exécution spéculative concourants, ledit compteur d'entrées (130) comprenant en outre N registres de décalage de secours (140, 142).

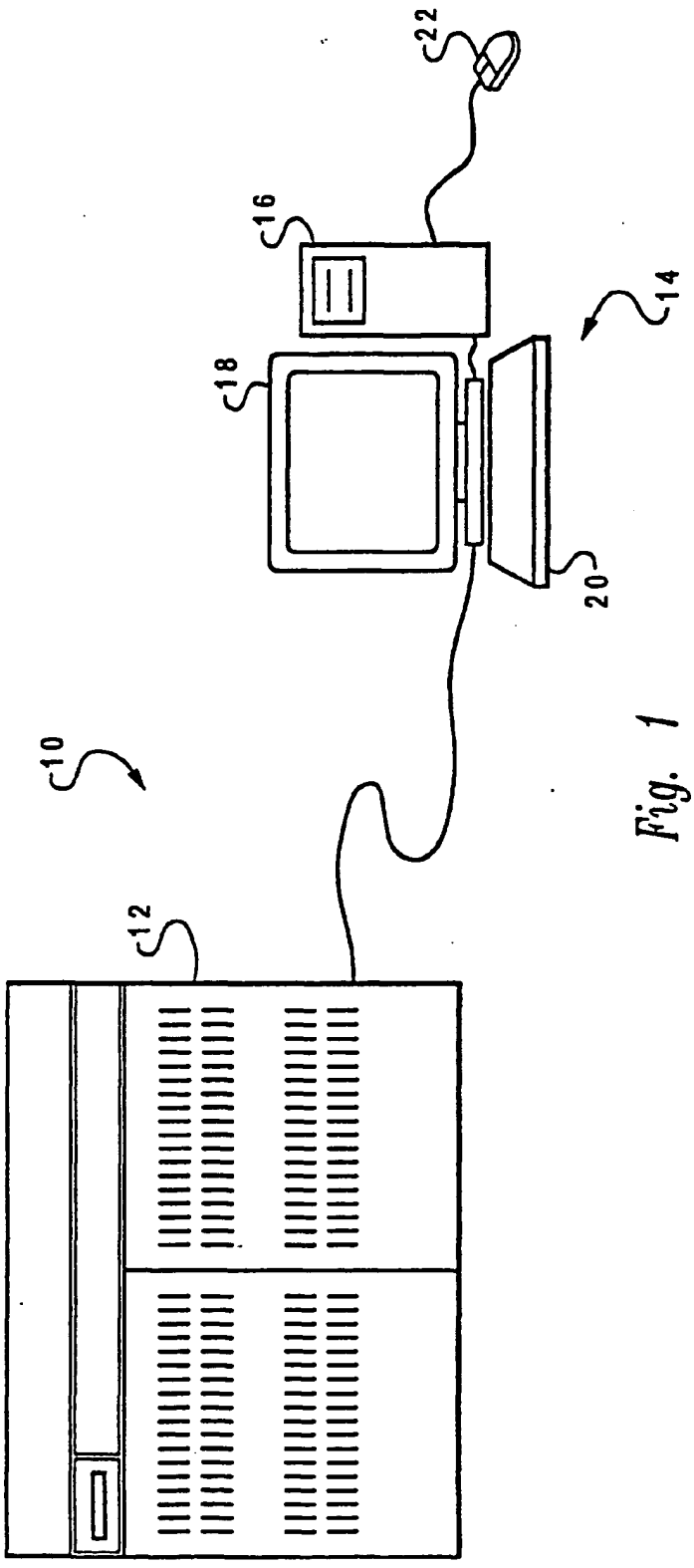

Fig. 1

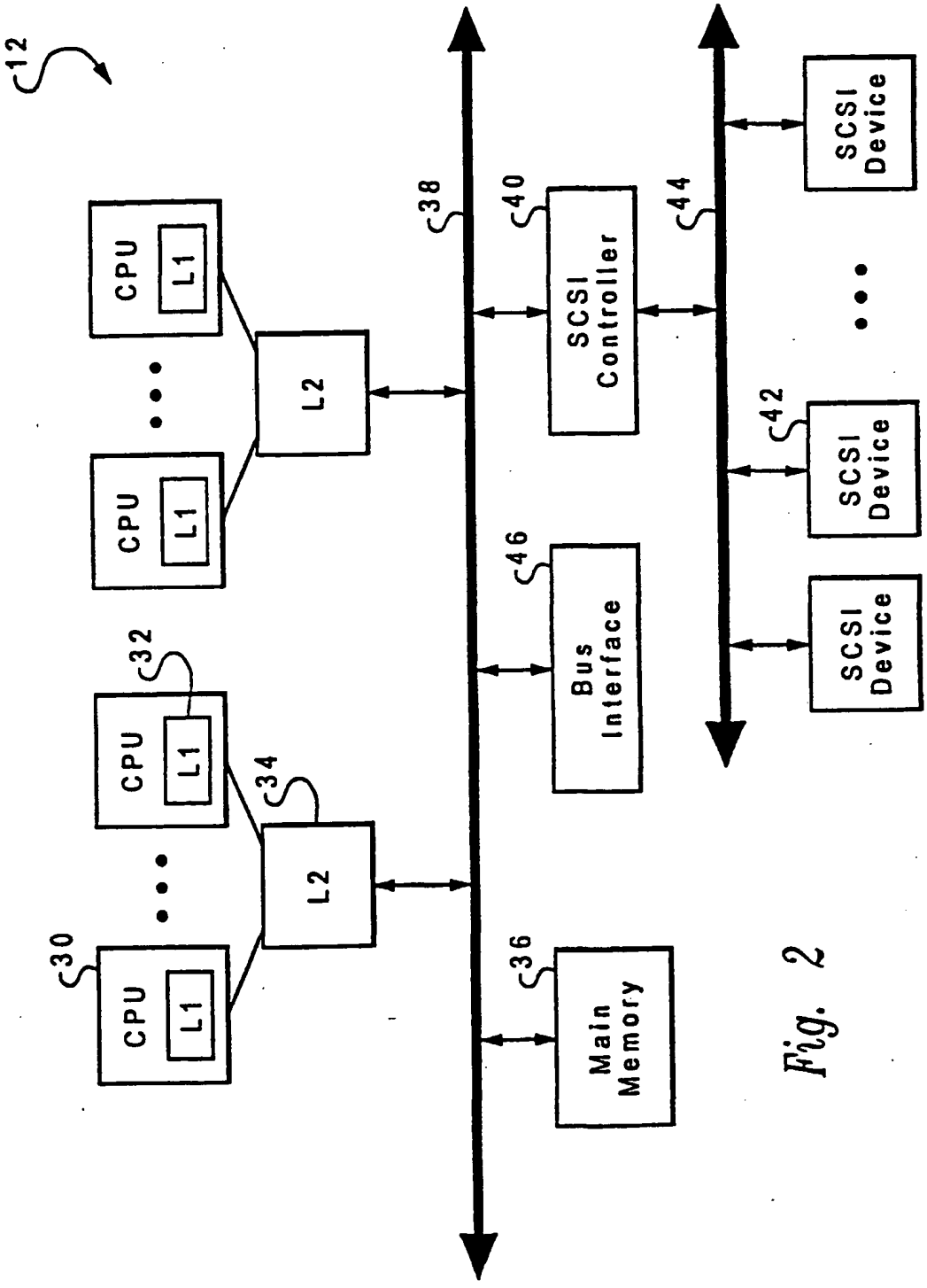
12
CPU
L1
L2
30
32
34
38
40
42
44
46
36
SCSI Controller
Bus Interface
Main Memory
SCSI Device

Fig. 2

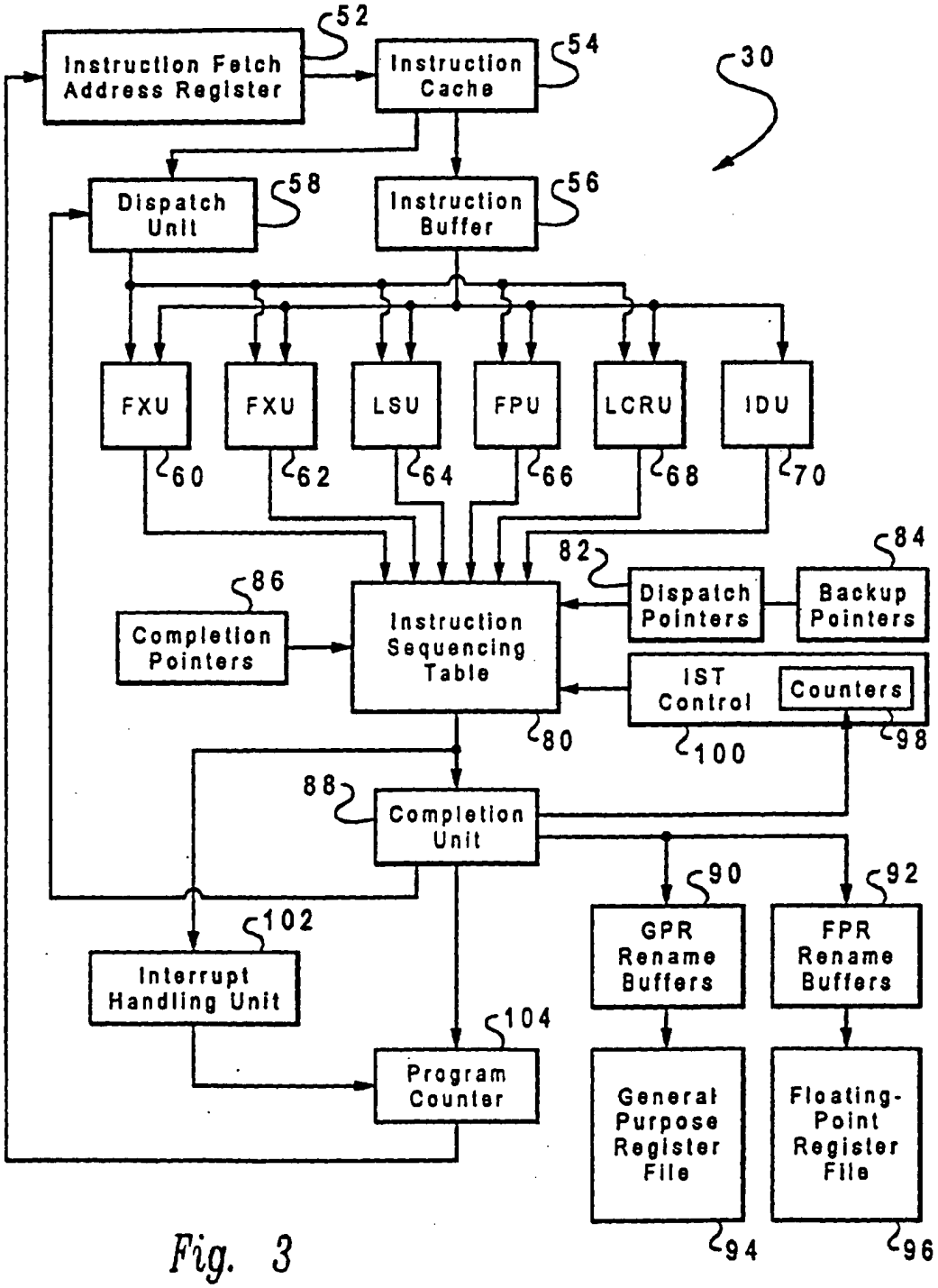
Instruction Fetch Address Register
52
Instruction Cache
54
30
Dispatch Unit
58
Instruction Buffer
56
FXU
FXU
LSU
FPU
LCRU
IDU
60
62
64
66
68
70
82
84
Dispatch Pointers
Backup Pointers
86
Completion Pointers
Instruction Sequencing Table
IST Control
Counters
80
100
98
88
Completion Unit
90
92
GPR Rename Buffers
FPR Rename Buffers
102
Interrupt Handling Unit
104
Program Counter
General Purpose Register File
Floating-Point Register File
94
96

Fig. 3

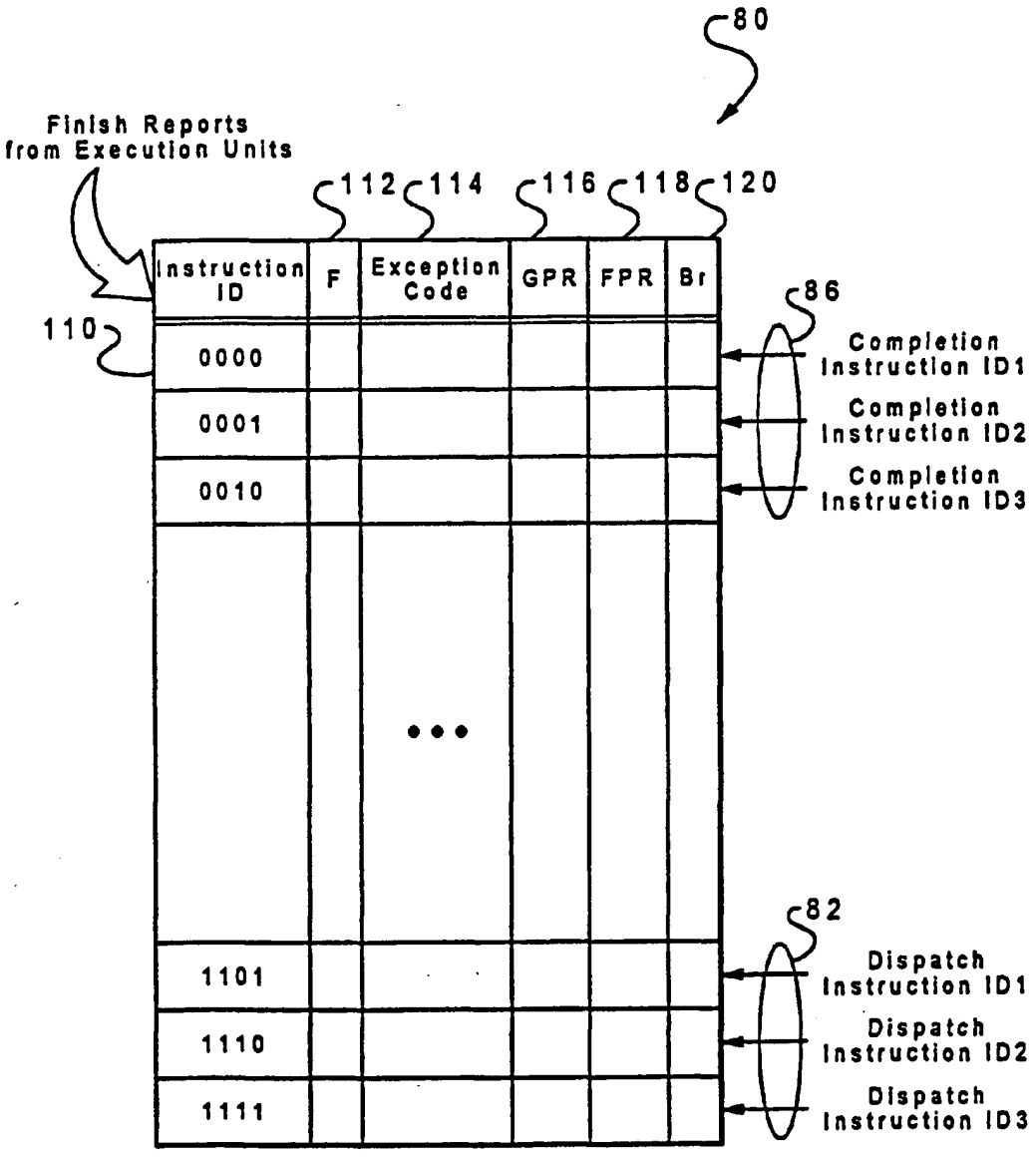
80
Finish Reports from Execution Units
112
114
116
118
120
Instruction ID
F
Exception Code
GPR
FPR
Br
110
86
0000
0001
0010
Completion Instruction ID1
Completion Instruction ID2
Completion Instruction ID3
82
1101
1110
1111
Dispatch Instruction ID1
Dispatch Instruction ID2
Dispatch Instruction ID3

Fig. 4

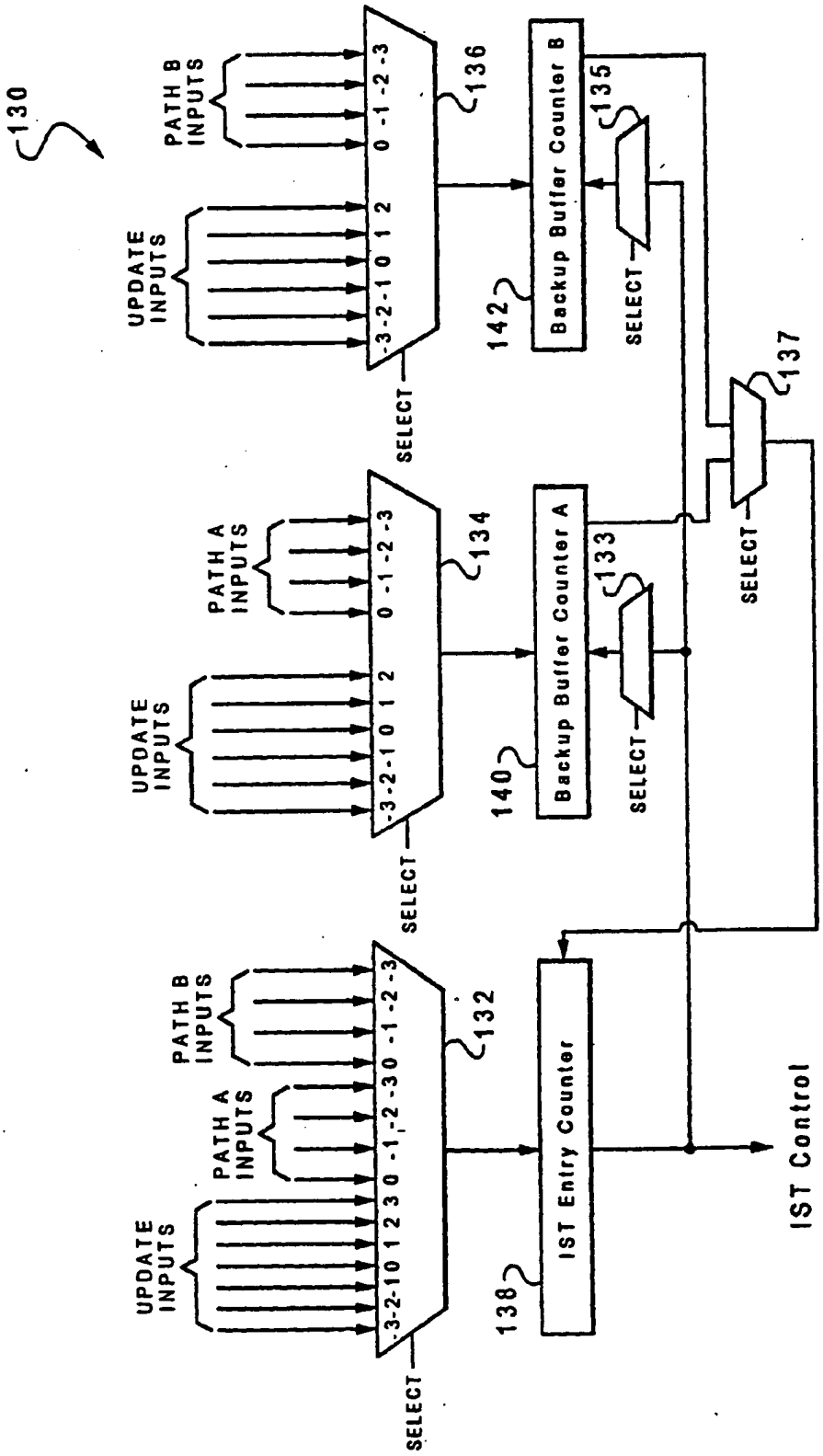
130
UPDATE INPUTS
PATH A INPUTS
PATH B INPUTS
-3 -2 -1 0 1 2 3
132
134
136
SELECT
IST Entry Counter
138
Backup Buffer Counter A
140
Backup Buffer Counter B
142
133
135
137
IST Control

Fig. 5

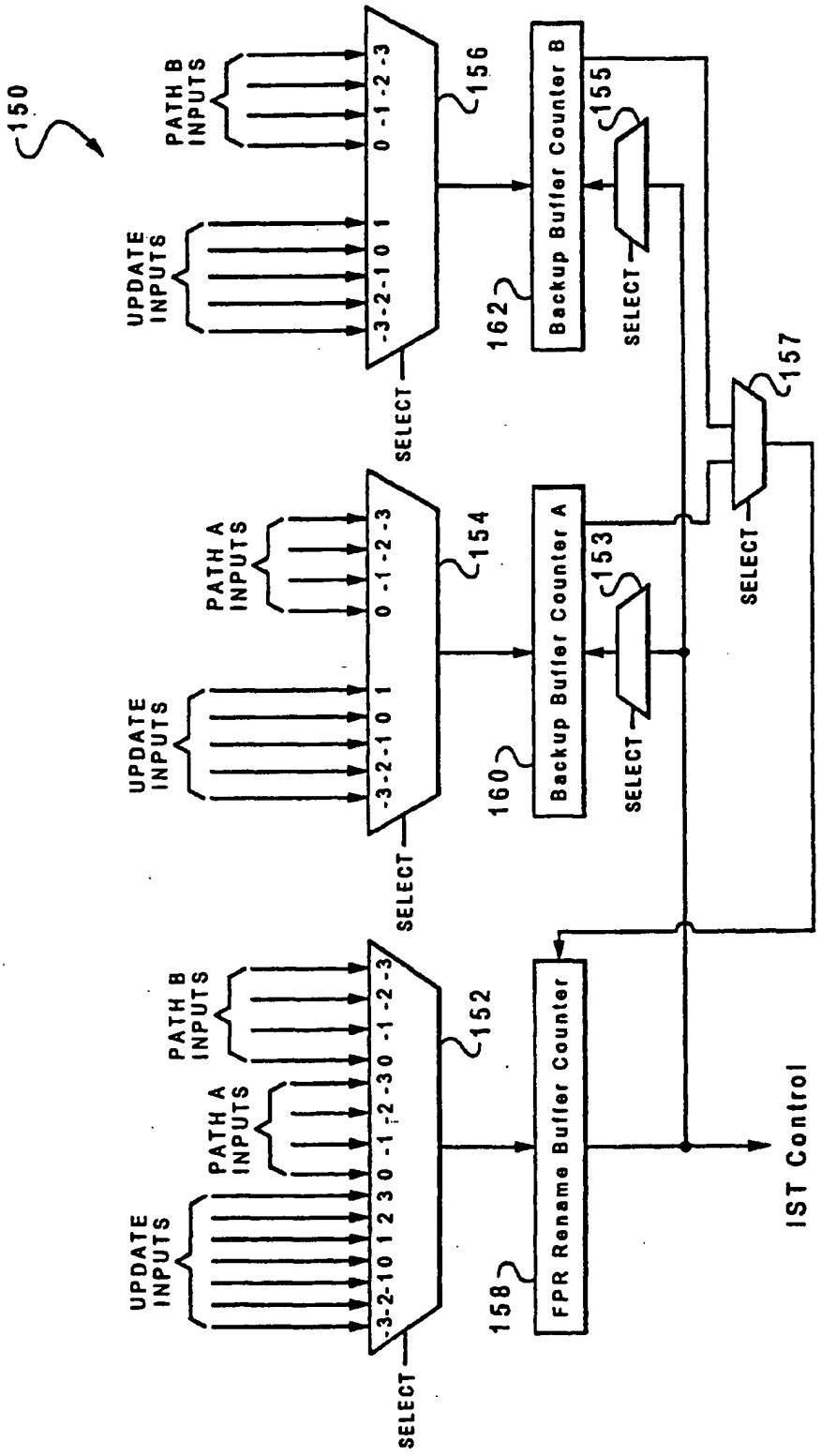
150
UPDATE INPUTS
PATH A INPUTS
PATH B INPUTS
-3 -2 -1 0 1 2 3 0 -1 -2 -3
SELECT
152
154
156
158
160
162
153
155
157
FPR Rename Buffer Counter
Backup Buffer Counter A
Backup Buffer Counter B
IST Control

Fig. 6

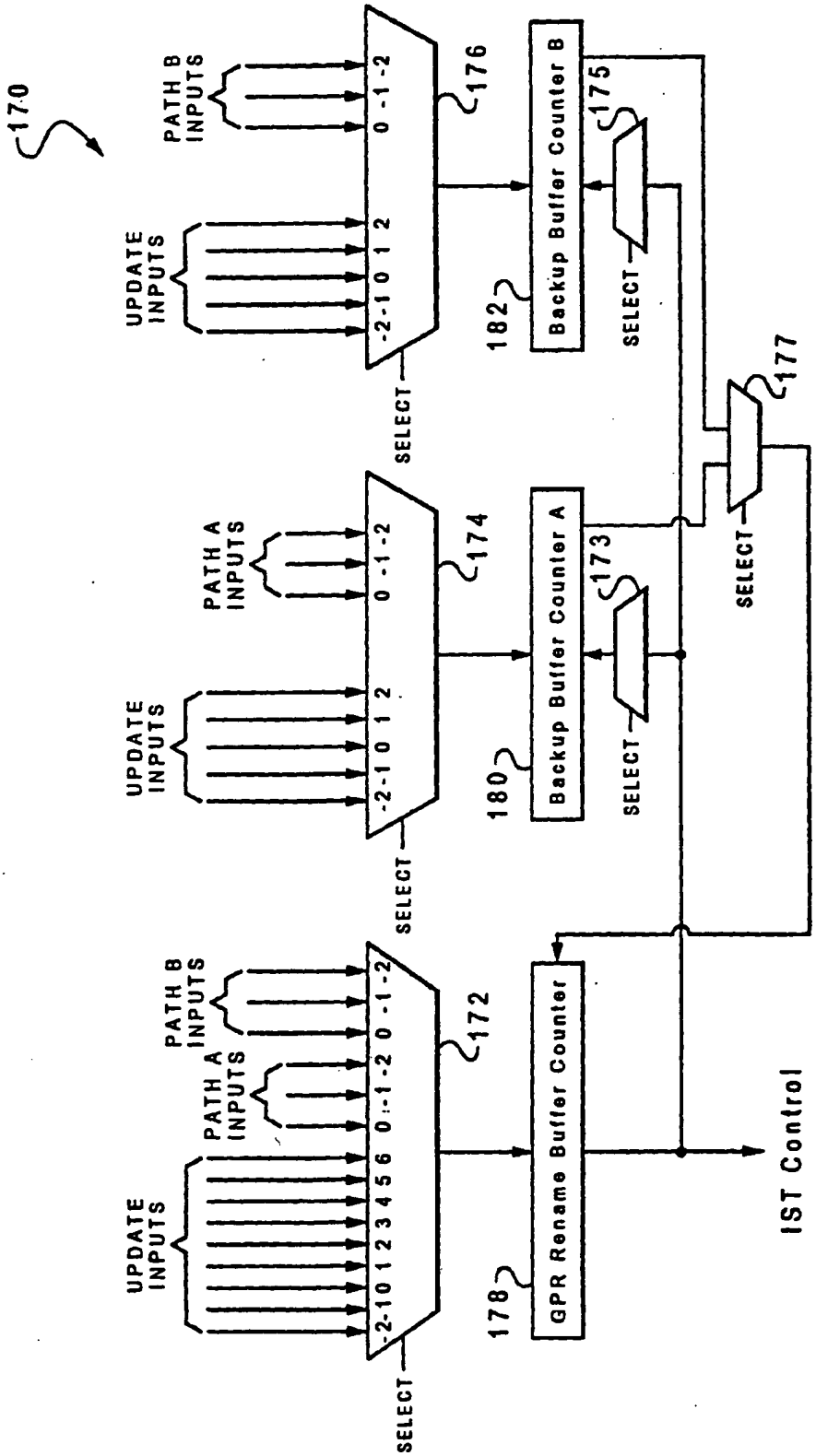
170
UPDATE INPUTS
PATH A INPUTS
PATH B INPUTS
-2 -1 0 1 2 3 4 5 6
0 -1 -2
172
SELECT
178
GPR Rename Buffer Counter
IST Control
174
180
Backup Buffer Counter A
173
176
182
Backup Buffer Counter B
175
177

Fig. 7

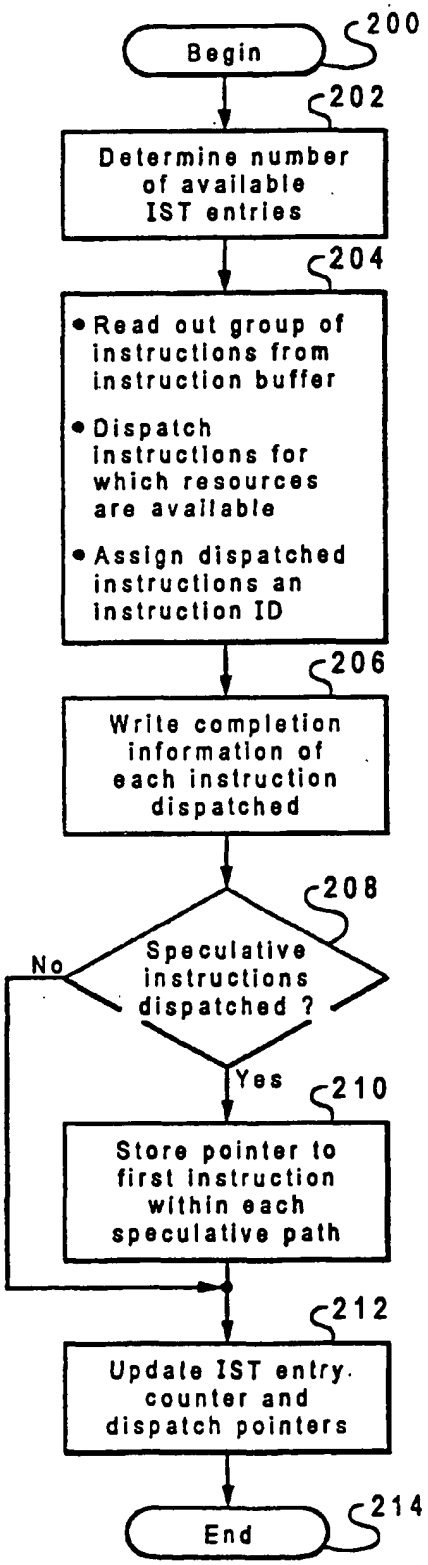
Begin
200
202
Determine number of available IST entries
204
• Read out group of instructions from instruction buffer
• Dispatch instructions for which resources are available
• Assign dispatched instructions an instruction ID
206
Write completion information of each instruction dispatched
208
Speculative instructions dispatched ?
No
Yes
210
Store pointer to first instruction within each speculative path
212
Update IST entry counter and dispatch pointers
End
214

Fig. 8

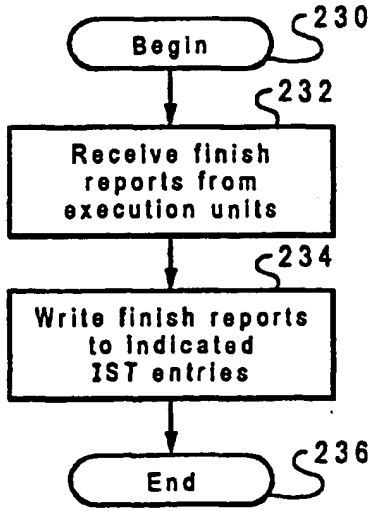
Begin
230
232
Receive finish reports from execution units
234
Write finish reports to indicated IST entries
End
236

Fig. 9

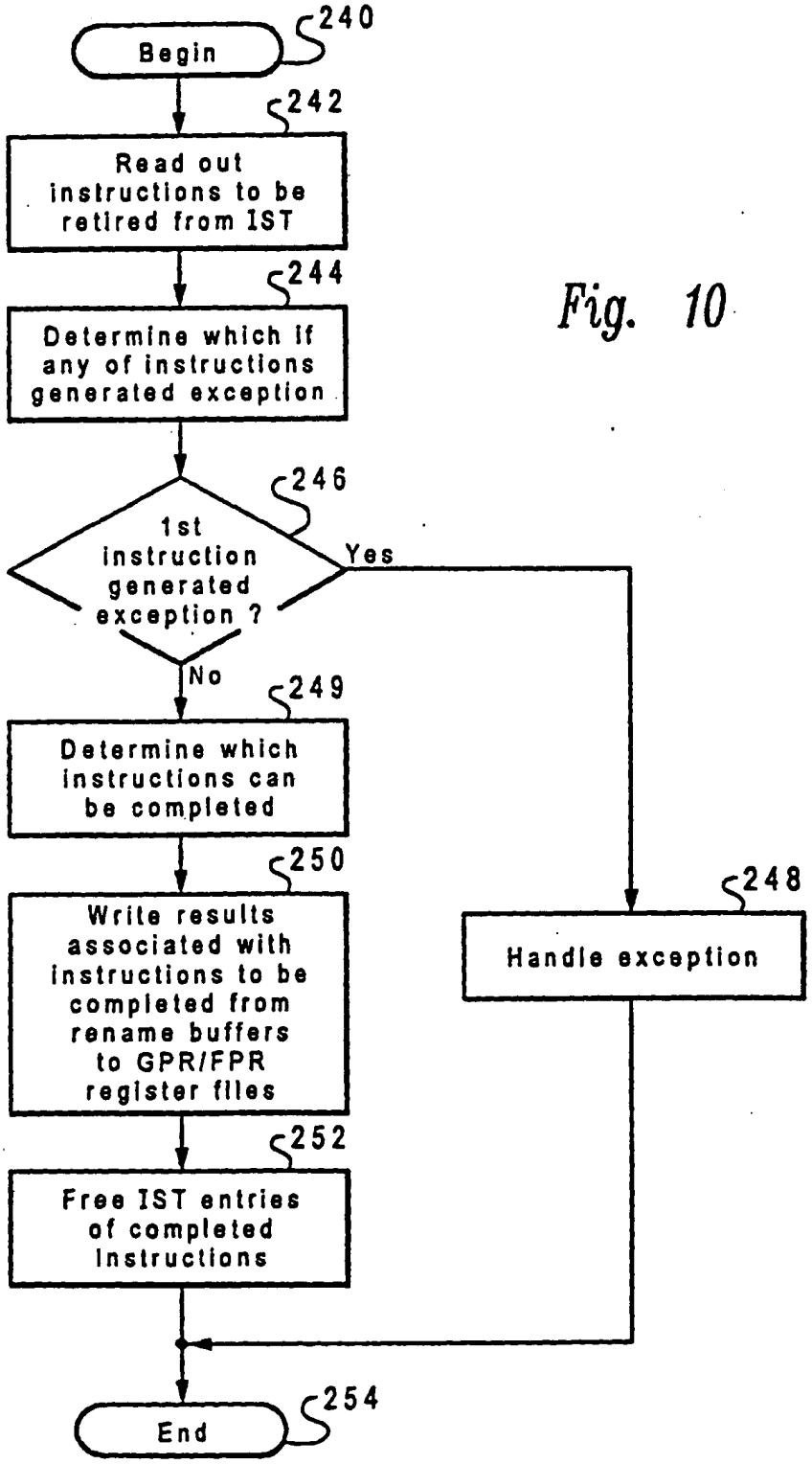
240
Begin
242
Read out instructions to be retired from IST
244
Determine which if any of instructions generated exception
246
1st instruction generated exception ?
Yes
No
249
Determine which instructions can be completed
250
Write results associated with instructions to be completed from rename buffers to GPR/FPR register files
252
Free IST entries of completed instructions
248
Handle exception
254
End

Fig. 10